Claims

1. An apparatus, comprising:

a computer readable medium storing code for a functional checker component for an Intellectual Property (IP) core;

a protocol checker component;

an interconnect monitor component; and

a programming interface to connect the functional checker component and the protocol checker component to the interconnect monitor component, wherein the interconnect monitor component having code to build data structures containing protocol data types requested by a first checker component and code specifying where to deliver data based upon a particular type of data requested by the first checker component.

2. The apparatus of claim 1, wherein code of the first checker component is the functional checker component for an IP core.

3. The apparatus of claim 1, wherein code of the first checker component is the protocol checker component.

4. The apparatus of claim 1, wherein the first checker component at the start of a simulation run directs the monitor component on the type of data requested and the locations to send that data in order to verify a hardware interconnect.

5. The apparatus of claim 1, wherein the software programming interface is coded in an object-oriented programming language.

6. The apparatus of claim 5, wherein the software programming interface is coded in C++.

7. The apparatus of claim 1, wherein the monitor component to create data structures that use shared pointers to pass data from a single monitor component to two or more checker components.

8. The apparatus of claim 1, wherein the monitor component contains a C++ template abstraction sensitive to the particular data type being requested by the first checker component when the first checker component registers a data callback service with the monitor component.

9. The apparatus of claim 8, wherein the particular data from the monitor component is available for analysis during a simulation run.

10. The apparatus of claim 1, wherein the code for the monitor component is separate from the code for the first checker component receiving data from the monitor component.

11. The apparatus of claim 1, wherein instances of the protocol data type are self maintaining via a reference counted object.

12. A machine-readable medium having stored thereon information representing the apparatus of claim 1.

13. The apparatus of claim 1, wherein the IP cores are components of a System on a Chip.

14. An apparatus, comprising:

a software programming interface to connect one or more functional checker components for an IP core and one or more protocol checker components to one or more interconnect monitor components, wherein the monitor components have code to create protocol data objects that track within

themselves whether their data is still being used by another checker component and when their protocol data object should be deleted; and

a computer readable medium to store the software programming interface.

15. The apparatus of claim 14, wherein the monitor component calls a method defined in the code of the checker component as data becomes available during the simulation run to allow an error to be detected during a simulation run.

16. The apparatus of claim 14, wherein the code for the monitor component is separate from the code for a first checker component receiving data from the monitor component.

17. The apparatus of claim 14, further comprising

two or more checker components to verify an interconnect protocol, wherein the monitor component supplies protocol data to the two or more checker components.

18. The apparatus of claim 14, wherein the monitor component contains data structures that use shared pointers to pass a single item of data from a monitor component to two or more locations within a single checker component.

19. The apparatus of claim 14, wherein the monitor component contains a C++ template abstraction sensitive to the particular data type being requested by a first checker component when the first checker component registers a data callback service with the monitor component.

20. The apparatus of claim 14, wherein a base class of the programming interface allows implementation across two or more types of interconnect protocols.

21. The apparatus of claim 14, wherein the IP cores are components of a System on a Chip.

22. The apparatus of claim 14, wherein the monitor component supplies callback services that deliver data requested by a checker component to one or more locations in the checker component based upon the particular data type requested by the checker component.

23. The apparatus of claim 14, wherein the another checker component is a functional checker component for an IP core.

24. The apparatus of claim 14, wherein the another checker component is a protocol checker.

25. The apparatus of claim 14, wherein a checker component at the start of a simulation run directs the monitor component on the type of data requested and the locations to send that data in order to verify a hardware interconnect.

26. The apparatus of claim 14, wherein the software programming interface is coded in an object-oriented programming language.

27. The apparatus of claim 14, wherein the software monitor component is programmed to collect data from hardware interconnect and to make the data available to a checker component to detect errors in an IP core design and communicate those errors to a user during a simulation run.

28. The apparatus of claim 14, further comprising:

a plurality of checker components, wherein at least one checker component models the functionality of an IP core using the data from the monitor component as input.

29. The apparatus of claim 14, wherein the programmer interface component includes a set of base classes for the data structures and monitor component callback services.

30. The apparatus of claim 14, wherein the functional checker component is used to detect errors in an IP core design and communicate those errors to a user during a simulation run.

31. The apparatus of claim 14, further comprising:

a protocol checker generation component to loop through all of the hardware interconnects in a design and to create an instance of an appropriate protocol checker for that connection.

32. The apparatus of claim 14, wherein a first monitor component to generate data structures as needed to supply data to a first checker component and a sum total of requests for data by the first checker component to a monitor component is not a maximal set of data provided by the first monitor component.

33. A machine-readable medium having stored thereon information representing the apparatus of claim 14.

34. A method, comprising:

monitoring a hardware interconnect between two or more IP cores to collect protocol data;

generating an object oriented data structure to pass protocol data to two or more locations associated with checker components; and

calling methods defined in the code of the checker components as data becomes available during a simulation run.

35. The method of claim 34, further comprising:

receiving a type of data requested and a location to send that data from the checker at the start of the simulation run.

36. The method of claim 34, further comprising:

providing callback services to the checker component at registration.

37. The method of claim 34, further comprising:

generating a data item that is self maintaining and will delete itself once all checker components are no longer referencing that data item.

38. The method of claim 34 further comprising:

using data structures that use shared pointers to pass data to locations in two or more checker components.

39. The method of claim 34, further comprising:

using data structures that use shared pointers to pass data to two or more locations within a checker component.

40. An apparatus, comprising:

means for monitoring a hardware interconnect between two or more IP cores to collect protocol data;

means for generating an object oriented data structure to pass protocol data to two or more locations associated with checker components; and

means for calling methods defined in the code of the checker components as data becomes available during a simulation run.

41. The apparatus of claim 40, further comprising:

means for receiving a type of data requested and a location to send that data from the checker anytime during the simulation run.

42. The apparatus of claim 40, further comprising:

means for providing callback services to the checker component at registration.

43. The apparatus of claim 40, further comprising:

means for generating a data item that is self maintaining and will delete itself once all checker components are no longer referencing that data item.

44. The apparatus of claim 40, further comprising:

means for using data structures that use shared pointers to pass data to locations in two or more checker components.

45. A machine-readable medium that provides instructions, which when executed by a machine, cause the machine to perform operations comprising:

collecting protocol data by monitoring an interconnect with an interconnect monitor component;

connecting, via a programming interface, one or more functional checker components for an IP core and one or more protocol checker components to the interconnect monitor component, wherein the monitor component has code to build data structures containing protocol data types requested by a first checker component and code on where to deliver data based upon a particular type of data requested by the first checker component.

46. The machine-readable medium of claim 45, further comprising instructions which, when executed by the machine, cause the machine to perform the further operations comprising:

generating an object oriented data structure to pass the protocol data to two or more locations associated with checker components.

47. The machine-readable medium of claim 45, further comprising instructions which, when executed by the machine, cause the machine to perform the further operations comprising:

generating a data item that is self maintaining and will delete itself once all checkers are no longer referencing that data item.